Fast algorithms for computing isogenies between elliptic curves

Alin Bostan, Bruno Salvy, Francois Morain, Eric Schost

Fast algorithms for computing isogenies between elliptic curves *

Alin Bostan, Bruno Salvy, Projet ALGO, INRIA Rocquencourt
Domaine de Voluceau, 78153 Le Chesnay Cedex, FRANCE
{ Alin.Bostan, Bruno.Salvy } @inria.fr

François Morain, Éric Schost, LIX, École polytechnique
91128 Palaiseau, France
{ Francois.Morain, Eric.Schost } @lix.polytechnique.fr

September 6, 2006

Abstract

We survey algorithms for computing isogenies between elliptic curves defined over a field of characteristic either 0 or a large prime. We introduce a new algorithm that computes an isogeny of degree ℓ (ℓ different from the characteristic) in time quasi-linear with respect to ℓ. This is based in particular on fast algorithms for power series expansion of the Weierstrass $\wp$-function and related functions.

1 Introduction

In the Schoof-Elkies-Atkin algorithm (SEA) that computes the cardinality of an elliptic curve over a finite field, isogenies between elliptic curves are used in a crucial way (see for instance [5] and the references we give later on). Isogenies have also been used to compute the ring of endomorphisms of a curve [34] and isogenies of small degrees play a role in [24, 17]. More generally, in various contexts, their computation becomes a basic primitive in cryptology (see [25, 9, 51, 20, 30, 53, 41]).

An important building block in Elkies's work is an algorithm that computes curves that are isogenous to a given curve E. This block uses modular polynomials to get the list of isogenous curves and Vélu's formulas to get the explicit form of the isogeny $I : E \to \tilde{E}$, where $\tilde{E}$ is in a suitable Weierstrass form.

In this work, we concentrate on algorithms that build the degree ℓ isogeny I from E and $\tilde{E}$ (and possibly some other parameters, see below). For the special case $\ell = 2$, formulas exist [49]; see also [16]. We could restrict further to the case when ℓ is an odd prime, since isogenies can be written as compositions of isogenies of prime degree, the

*This work was supported in part by the French National Agency for Research (ANR Gecko).

case of prime powers using isogeny cycles [18, 16, 23]. Besides, the odd prime case is the most important one in SEA. However, our results stand for arbitrary ℓ.

We demand that the characteristic p of the base field $\mathbf{K}$ be 0 or $p \gg \ell$. This restriction is satisfied in the case of interest in the application to the SEA algorithm, since otherwise p-adic methods are much faster and easier to use [42, 33]. Several approaches to isogeny computation are available in small characteristic: we refer to [13, 38] for an approach via formal groups, [36] for the special case $p = 2$, and [14, 15, 37, 31] for the general case of p small. The case of $p = \ell$ deserves a special treatment, see [14, 38], using Gunji's work [27] as main ingredient (see also [37]).

Our assumption on p implies that the equations of our curves can be written in the Weierstrass form

$$y^2 = x^3 + Ax + B. \tag{1}$$

In characteristic zero, the curve (1) can be parameterized by $(x, y) = (\wp(z), \wp'(z)/2)$ in view of the classical differential equation

$$\wp'(z)^2 = 4(\wp(z)^3 + A\wp(z) + B) \tag{2}$$

satisfied by the Weierstrass $\wp$-function. This is the basis for our computation of isogenies. We thus prove two results, first on the computation of the Weierstrass $\wp$-function, and then on the computation of the isogeny itself.

Our main contribution is to exploit classical fast algorithms for power series computations and show how they apply to the computation of isogenies. We denote by $\mathsf{M} : \mathbb{N} \to \mathbb{N}$ a function such that polynomials of degree less than n can be multiplied in $\mathsf{M}(n)$ base field operations. Using the fast Fourier transform [44, 10], one can take $\mathsf{M}(n) \in O(n \log n \log\log n)$; over fields containing primitive roots of unity, one can take $\mathsf{M}(n) \in O(n \log n)$. We make the standard super-linearity assumptions on the function M, see the following section.

Theorem 1. *Let $\mathbf{K}$ be a field of characteristic zero. Given A and B in $\mathbf{K}$, the first n coefficients of the Laurent expansion at the origin of the function $\wp$ defined by (2) can be computed in $O(\mathsf{M}(n))$ operations in $\mathbf{K}$.*

In §3, we give a more precise version of this statement, that handles the case of fields of positive, but large enough, characteristic.

An isogeny is a regular map between two elliptic curves that is also a group morphism. If E and $\tilde{E}$ are in Weierstrass form and $I = (I_x, I_y)$ is an isogeny $E \to \tilde{E}$, then $I_x(P)$ depends only on the x-coordinate of P, and there exists a constant $c \in \mathbf{K}$ such that $I_y = cyI'_x$. Following Elkies [21, 22], we consider only so-called *normalized* isogenies, those for which $c = 1$ (such isogenies are used for instance in SEA). In this case, we will write σ for the sum of the abscissas of non-zero points in the kernel of I.

Theorem 2. *Let $\mathbf{K}$ be a field of characteristic p and let E and $\tilde{E}$ be two curves in Weierstrass form, such that there exists a normalized isogeny $I : E \to \tilde{E}$ of degree ℓ. Then, one can compute the isogeny I*

1. *in $O(\mathsf{M}(\ell))$ operations in $\mathbf{K}$, if $p = 0$ or $p > 2\ell - 1$, if σ is known;*
2. *in $O(\mathsf{M}(\ell) \log \ell)$ operations in $\mathbf{K}$, if $p = 0$ or $p > 8\ell - 5$, without prior knowledge of σ.*

Taking $\mathsf{M}(n) \in O(n \log n \log\log n)$ shows that the complexity results in Theorems 1 and 2 are nearly optimal, up to polylogarithmic factors. Notice that the algorithms using modular equations to detect isogenies yield the value of σ as a by-product. However, in a cryptographic context, this may not be the case anymore; this is why we distinguish the two cases in Theorem 2.

This article is organized as follows. In §2, we recall known results on the fast computation of truncated power series, using notably Newton's iteration. In §3, we show how these algorithms apply to the computation of the $\wp$-function. Then in §4, we recall the definition of isogenies and the properties we need, and give our quasi-linear algorithms; examples are given in §5. In the next section, we survey previous algorithms for the computation of isogenies. Their complexity has not been discussed before; we analyze them when combined with fast power series expansions so that a comparison can be made. Finally, in §7, we report on our implementation.

2 A review of fast algorithms for power series

The algorithms presented in this section are well-known; they reduce several problems for power series (reciprocal, exponentiation, ...) to polynomial multiplication.

Our main tool to devise fast algorithms is Newton's iteration; it underlies the $O(\mathsf{M}(\ell))$ result reported in Theorem 1, and in the (practically important) point (1) of Theorem 2. Hence, this question receives most of our attention below, with detailed pseudo-code. We will be more sketchy on some other algorithms, such as rational function reconstruction, referring to the relevant literature.

We suppose that the *multiplication time* function M is super-linear, *i.e.*, it satisfies the following inequality (see, e.g., [26, Chapter 8]):

$$\frac{\mathsf{M}(n)}{n} \leq \frac{\mathsf{M}(n')}{n'} \qquad \text{if } n \leq n'. \tag{3}$$

In particular, Equation (3) implies the inequality

$$\mathsf{M}(1) + \mathsf{M}(2) + \mathsf{M}(4) + \cdots + \mathsf{M}(2^i) \leq 2\mathsf{M}(2^i),$$

which is the key to show that all algorithms based on Newton's iteration have complexity in $O(\mathsf{M}(n))$. Cantor and Kaltofen [10] have shown that one can take $\mathsf{M}(n)$ in $O(n \log n \log\log n)$; as a byproduct, most questions addressed below admit similar quasi-linear estimates.

2.1 Reciprocal

Let $f = \sum_{i\geq 0} f_i z^i$ be in $\mathbf{K}[[z]]$, with $f_0 \neq 0$, and let $g = 1/f = \sum_{i\geq 0} g_i z^i$ in $\mathbf{K}[[z]]$. The coefficients g_i can be computed iteratively by the formula

$$g_0 = \frac{1}{f_0} \quad \text{and} \quad g_i = -\frac{1}{f_0} \sum_{j=1}^{i} f_j g_{i-j} \quad \text{for} \quad i \geq 1.$$

For a general f, the cost of computing $1/f \bmod z^n$ with this method is in $O(n^2)$; observe nevertheless that if f is a polynomial of degree d, the cost reduces to $O(nd)$.

To speed up the computation in the general case, we use Newton's iteration. For reciprocal computation, it amounts to computing a sequence of truncated power series h_i as follows:

$$h_0 = \frac{1}{f_0} \quad \text{and} \quad h_{i+1} = h_i(2 - fh_i) \bmod z^{2^{i+1}} \text{ for } i \geq 0.$$

Then, $h_i = 1/f \bmod z^{2^i}$. As a consequence, $1/f \bmod z^n$ can be computed in $O(\mathsf{M}(n))$ operations. This result is due to Cook for an analogous problem of integer inversion [12], and to Sieveking [48] and Kung [35] in the power series case.

2.2 Exponentiation

Let f be in $\mathbf{K}[[z]]$, with $f(0) = 0$. Given n in $\mathbb{N}$, such that $2, \ldots, n-1$ are units in $\mathbf{K}$, the truncated exponential $\exp_n(f)$ is defined as

$$\exp_n(f) = \sum_{i=0}^{n-1} \frac{1}{i!} f^i \bmod z^n.$$

Conversely, if g is in $1 + z\mathbf{K}[[z]]$, its truncated logarithm is defined as

$$\log_n(g) = -\sum_{i=1}^{n-1} \frac{1}{i}(1-g)^i \bmod z^n.$$

The truncated logarithm is obtained by computing the Taylor expansion of g'/g modulo z^{n-1} using the algorithm of the previous subsection, and taking its antiderivative; hence, it can be computed in $O(\mathsf{M}(n))$ operations.

Building on this, Brent [6] introduced the Newton iteration

$$g_0 = 1, \quad g_{i+1} = g_i(1 + f - \log_{2^{i+1}}(g_i)) \bmod z^{2^{i+1}}$$

to compute the sequence $g_i = \exp_{2^i}(f)$. As a consequence, $\exp_n(f)$ can be computed in $O(\mathsf{M}(n))$ operations as well, whereas the naive algorithm has cost $O(n^2)$.

As an application, Schönhage [43] gave a fast algorithm to recover a polynomial f of degree n from its first n power sums $p_1, \ldots, p_n$. Schönhage's algorithm is based on the fact that the logarithmic derivative of f at infinity is the generating series of its power sums, that is,

$$z^n f\left(\frac{1}{z}\right) = \exp_{n+1}\left(-\sum_{i=1}^{n} \frac{p_i}{i} z^i\right).$$

Hence, given $p_1, \ldots, p_n$, the coefficients of f can be recovered in time $O(\mathsf{M}(n))$. This algorithm requires that $2, \ldots, n$ be units in $\mathbf{K}$.

2.3 First-order linear differential equations

Let a, b, c be in $\mathbf{K}[[z]]$, with $a(0) \neq 0$, and let α be in $\mathbf{K}$. We want to compute the first n terms of $f \in \mathbf{K}[[z]]$ such that

$$af' + bf = c \quad \text{and} \quad f(0) = \alpha$$

Let $B = b/a \bmod z^{n-1}$ and $C = c/a \bmod z^{n-1}$. Then, defining $J = \exp_n(\int B)$, f satisfies the relation

$$f = \frac{1}{J}\left(\alpha + \int CJ\right) \bmod z^n.$$

Using the previous reciprocal and exponentiation algorithms, $f \bmod z^n$ can thus be computed in time $O(\mathsf{M}(n))$. This algorithm is due to Brent and Kung [8]; it requires that $2, \ldots, n-1$ be units in $\mathbf{K}$.

2.4 First-order nonlinear differential equations

We only treat this question in a special case, following again Brent and Kung's article [8, Theorem 5.1]. Let G be in $\mathbf{K}[[z]][t]$, let α, β be in $\mathbf{K}$, and let $f \in \mathbf{K}[[z]]$ be a solution of the equation

$$f'^2 = G(z, f), \quad f(0) = \alpha, \quad f'(0) = \beta,$$

with furthermore $\beta^2 = G(0, \alpha) \neq 0$. Supposing that, for $s \geq 2$, the initial segment $f_1 = f \bmod z^s$ is known, we show how to deduce $f \bmod z^{2s-1}$. Write $f = f_1 + f_2 \bmod z^{2s-1}$, where z^s divides f_2. One checks that f_2 is a solution of the linearized equation

$$2f_1'f_2' - G_t(z, f_1)f_2 = G(z, f_1) - f_1'^2 \mod z^{2s-2}, \tag{4}$$

with the initial condition $f_2(0) = 0$, where G_t denotes the derivative of G with respect to t. The condition $f'(0) \neq 0$ implies that f_1' is a unit in $\mathbf{K}[[z]]$; then, the cost of computing $f_2 \bmod z^{2s-1}$ is in $O(\mathsf{M}(s))$ (remark that we do not take the degree of G into account). Finally, the computation of f at precision n is as follows:

1. Let $f = \alpha + \beta z \bmod z^2$ and $s = 2$;
2. **while** $s < n$ **do**
   (a) Compute $f \bmod z^{2s-1}$ from $f \bmod z^s$;
   (b) Let $s = 2s - 1$.

Due to the super-linearity of M, $f \bmod z^n$ can thus be computed using $O(\mathsf{M}(n))$ operations. Again, we have to assume that $2, \ldots, n-1$ are units in $\mathbf{K}$.

2.5 Other algorithms.

We conclude this section by pointing out other algorithms that are used below.

Power series composition. Over a general field $\mathbf{K}$, there is no known algorithm of quasi-linear complexity for computing $f(g) \bmod z^n$, for f, g in $\mathbf{K}[[z]]$. The best results known today are due to Brent and Kung [8]. Two algorithms are proposed in that article, of respective complexities $O(\mathsf{M}(n)\sqrt{n} + n^{\frac{\omega+1}{2}})$ and $O(\mathsf{M}(n)\sqrt{n \log n})$, where $2 \leq \omega < 3$ is the exponent of matrix multiplication (see, e.g., [26, Chapter 12]). Over fields of positive characteristic p, Bernstein's algorithm for composition [3] has complexity $O(\mathsf{M}(n))$, but the $O(\,)$ estimate hides a linear dependence in p, making it inefficient in our setting ($p \gg n$).

Rational function reconstruction. Our last subroutine consists in reconstructing a rational function from its Taylor expansion at the origin. Suppose that f is in $\mathbf{K}(z)$ with numerator and denominator of degree bounded respectively by n and n', and with denominator non-vanishing at the origin; then, knowing the first $n+n'+1$ terms of the expansion of f at the origin, the rational function f can be reconstructed in $O(\mathsf{M}(n+n')\log(n+n'))$ operations, see [7].

3 Computing the Weierstrass $\wp$-function

3.1 The Weierstrass $\wp$-function

We now study the complexity of computing the Laurent series expansion of the Weierstrass $\wp$-function at the origin, thus proving Theorem 1. We suppose for a start that the base field $\mathbf{K}$ equals $\mathbb{C}$; the positive characteristic case is discussed below. Let thus A, B be in $\mathbf{K} = \mathbb{C}$. The Weierstrass function $\wp$ associated to A and B is a solution of the non-linear differential equation (2); its Laurent expansion at the origin has the form

$$\wp(z) = \frac{1}{z^2} + \sum_{i \geq 1} c_i z^{2i}. \tag{5}$$

The goal of this section is to study the complexity of computing the first terms $c_1, \ldots, c_n$. We first present a "classical" algorithm, and then show how to apply the fast algorithms for power series of the previous section.

3.2 Quadratic algorithm

First, we recall the direct algorithm. Substituting the expansion (5) into Equation (2) and identifying coefficients of z^{-2} and z^0 gives

$$c_1 = -\frac{A}{5} \quad \text{and} \quad c_2 = -\frac{B}{7}.$$

Next, differentiating Equation (2) yields the second order equation

$$\wp'' = 6\wp^2 + 2A. \tag{6}$$

This equation implies that for $k \geq 3$, c_k is given by

$$c_k = \frac{3}{(k-2)(2k+3)} \sum_{i=1}^{k-2} c_i c_{k-1-i}. \tag{7}$$

Hence, the coefficients $c_1, \ldots, c_n$ can be computed using $O(n^2)$ operations in $\mathbf{K}$.

If the characteristic p of $\mathbf{K}$ is positive, the definition of $\wp$ as a Laurent series fails, due to divisions by zero. However, assuming $p > 2n+3$, it is still possible to define the coefficients $c_1, \ldots, c_n$ through the previous recurrence relation. Then, again, $c_1, \ldots, c_n$ can be computed using $O(n^2)$ operations in $\mathbf{K}$.

3.3 Fast algorithm

We first introduce new quantities, that are used again in the next section. Define

$$Q(z) = \frac{1}{\wp(z)} \in z^2 + z^6\mathbf{K}[[z^2]] \quad \text{and} \quad R(z) = \sqrt{Q(z)} \in z + z^5\mathbf{K}[[z^2]].$$

The differential equation satisfied by R is

$$R'(z)^2 = B\,R(z)^6 + A\,R(z)^4 + 1, \tag{8}$$

from which we can deduce the first terms of R:

$$R(z) = z + \frac{A}{10}z^5 + \frac{B}{14}z^7 + O(z^8) = z\left(1 + \frac{A}{10}z^4 + \frac{B}{14}z^6 + O(z^7)\right).$$

Squaring R yields

$$Q(z) = z^2 + \frac{A}{5}z^6 + \frac{B}{7}z^8 + O(z^9) = z^2\left(1 + \frac{A}{5}z^4 + \frac{B}{7}z^6 + O(z^7)\right).$$

Taking the reciprocal of the right-hand series finally yields

$$\wp(z) = \frac{1}{z^2}\left(1 - \frac{A}{5}z^4 - \frac{B}{7}z^6 + O(z^7)\right) = \frac{1}{z^2} - \frac{A}{5}z^2 - \frac{B}{7}z^4 + O(z^5),$$

as requested. Thus, our fast algorithm to compute the coefficients $c_1, \ldots, c_n$ is as follows:

1. Compute $R(z) \bmod z^{2n+4}$ using the algorithm of §2.4 with $G = Bt^6 + At^4 + 1$;
2. Compute $Q(z) = R(z)^2 \bmod z^{2n+5}$;
3. Compute $\wp(z) = 1/Q(z) \bmod z^{2n+1}$.

In the first step, we remark that our assumption $R'(0) \neq 0$ is indeed satisfied, hence $R(z) \bmod z^{2n+4}$ can be computed in $O(\mathsf{M}(n))$ operations, assuming $2, \ldots, 2n+3$ are units in $\mathbf{K}$. Using the algorithm of §2.1, the squaring and reciprocal necessary to recover $\wp(z) \bmod z^{2n+1}$ admit the same complexity bound. This proves Theorem 1.

4 Fast computation of isogenies

In this section, we recall the basic properties of isogenies and an algorithm due to Elkies [22] that computes an isogeny of degree ℓ in quadratic complexity $O(\ell^2)$. Then, we design two fast variants of Elkies' algorithm, by exploiting the differential equations satisfied by some functions related to the Weierstrass function, proving Theorem 2.

4.1 Isogenies

The following properties are classical; all the ones not proved here can be found for instance in [49, 50]. Let E and $\tilde{E}$ be two elliptic curves defined over $\mathbf{K}$. An isogeny between E and $\tilde{E}$ is a regular map $I : E \to \tilde{E}$ that is also a group morphism. Hence, we have $\tilde{E} \simeq E/F$, where F is the kernel of I; here, our isogenies are all non-zero.

The most elementary example of an isogeny is the "multiplication by m" map which sends $P \in E$ to $[m]P$, where, as usual, the group law on E is written additively. If E is given through a Weierstrass model, the group law yields the following formulas for $[m]P$ in terms of the *Weber polynomials* $\psi_m(x, y)$ [49, p. 105]:

$$[m](x, y) = \left(\frac{\phi_m(x, y)}{\psi_m(x, y)^2}, \frac{\omega_m(x, y)}{\psi_m(x, y)^3} \right). \tag{9}$$

Using the Weierstrass equation of E, the polynomial $\psi_m(x, y)$ rewrites in terms of the so-called *division polynomial* $f_m(x)$, which is univariate of degree $\Theta(m^2)$: $\psi_m(x, y) = f_m(x)$ if m is odd, $\psi_m(x, y) = 2y f_m(x)$ otherwise.

Given an isogeny $I : E \to \tilde{E}$, there exist a unique isogeny (the *dual isogeny*) $\hat{I} : \tilde{E} \to E$ and a unique integer ℓ such that $\hat{I} \circ I = [\ell]$; the integer ℓ is called the *degree* of I; if I is separable, it equals the cardinality of its kernel. For instance, the degree of the isogeny $[m]$ is m^2 and this is reflected by the degree of the division polynomials.

Let E and $\tilde{E}$ be two isogeneous elliptic curves in Weierstrass form, defined over $\mathbf{K}$. Then the isogeny I between E and $\tilde{E}$ can be written as

$$I(x, y) = (I_x(x), cyI'_x(x)), \tag{10}$$

for some c in $\mathbf{K}$. We say that I is *normalized* if the constant c equals 1; in this case, I is separable. We use an explicit form for such isogenies, extending results of Kohel [34, §2.4] and Dewaghe [19] to the case of arbitrary degree ℓ.

Proposition 4.1. *Let $I : E \to \tilde{E}$ be a normalized isogeny of degree ℓ and let F be its kernel. Then I can be written as*

$$I(x, y) = \left(\frac{N(x)}{D(x)}, y \left(\frac{N(x)}{D(x)} \right)' \right), \tag{11}$$

where D is the polynomial

$$D(x) = \prod_{Q \in F^*} (x - x_Q) = x^{\ell-1} - \sigma x^{\ell-2} + \sigma_2 x^{\ell-3} - \sigma_3 x^{\ell-4} + \cdots \tag{12}$$

and $N(x)$ is related to $D(x)$ through the formula

$$\frac{N(x)}{D(x)} = \ell x - \sigma - (3x^2 + A)\frac{D'(x)}{D(x)} - 2(x^3 + Ax + B)\left(\frac{D'(x)}{D(x)} \right)'. \tag{13}$$

Proof. Note first that given a subgroup F of $E(\overline{\mathbf{K}})$, there can exist only one pair $(\tilde{E}, I)$ where $\tilde{E}$ is in Weierstrass form and I is a normalized isogeny $E \to \tilde{E}$ having F as kernel.

In [54], Vélu constructs the curve $\tilde{E}$ and the normalized isogeny I, starting from the coordinates of the points in its kernel F. A point P of coordinates (x_P, y_P) is sent by the isogeny I to a point of coordinates

$$x_{I(P)} = x_P + \sum_{Q \in F^*} (x_{P+Q} - x_Q) \quad \text{and} \quad y_{I(P)} = y_P + \sum_{Q \in F^*} (y_{P+Q} - y_Q).$$

From there, Vélu uses the group law to get explicit expressions of the coordinates. More precisely, write F_2 for the set of points in F that are of order 2. Then F can be written as

$$F = \{O_E\} \cup F_2 \cup F_{\text{odd}} \cup (-F_{\text{odd}}),$$

where $F_{\text{odd}} \cap (-F_{\text{odd}}) = \emptyset$ and $-F_{\text{odd}}$ denotes the set of opposite points of F_{odd}, so that $D(x)$ rewrites as

$$D(x) = \prod_{Q \in F_2} (x - x_Q) \prod_{Q \in F_{\text{odd}}} (x - x_Q)^2.$$

Finally, let $F^+ = F_2 \cup F_{\text{odd}}$. Then Vélu gave the following explicit form for $I(x, y) = (I_x(x), yI'_x(x))$:

$$I_x(x) \;=\; x + \sum_{Q \in F^+} \left(\frac{t_Q}{x - x_Q} + 4 \frac{x_Q^3 + Ax_Q + B}{(x - x_Q)^2} \right),$$

where $t_Q = 3x_Q^2 + A$ if $Q \in F_2$ and $t_Q = 2(3x_Q^2 + A)$ otherwise. Observing that for $Q \in F_2$, $x_Q^3 + Ax_Q + B$ equals 0, one sees that I_x admits D for denominator, as claimed in Equation (11).

Next, we split the sum over F^+ into that for $Q \in F_2$ and that for $Q \in F_{\text{odd}}$. The former rewrites as

$$\sum_{Q \in F_2} \left(\frac{3x_Q^2 + A}{x - x_Q} + 2 \frac{x_Q^3 + Ax_Q + B}{(x - x_Q)^2} \right),$$

since these points satisfy $x_Q^3 + Ax_Q + B = 0$; the sum for $Q \in F_{\text{odd}}$ rewrites as

$$\frac{1}{2} \sum_{Q \in F_{\text{odd}} \cup -F_{\text{odd}}} \left(\frac{t_Q}{x - x_Q} + 4 \frac{x_Q^3 + Ax_Q + B}{(x - x_Q)^2} \right).$$

Therefore, we obtain

$$I_x(x) = x + \sum_{Q \in F^*} \left(\frac{3x_Q^2 + A}{x - x_Q} + 2 \frac{x_Q^3 + Ax_Q + B}{(x - x_Q)^2} \right),$$

which can be rewritten as

$$I_x(x) \;=\; x + \sum_{Q \in F^*} \left(x - x_Q - \frac{3x^2 + A}{x - x_Q} + 2 \frac{x^3 + Ax + B}{(x - x_Q)^2} \right).$$

This yields Equation (13). □

Though this is not required in what follows, let us mention how Vélu's formulæ enable one to construct the curve $\tilde{E}$. Let $\sigma, \sigma_2, \sigma_3$ be as in Equation (12) and

$$t = A(\ell - 1) + 3(\sigma^2 - 2\sigma_2),$$

$$w = 3A\sigma + 2B(\ell - 1) + 5(\sigma^3 - 3\sigma\sigma_2 + 3\sigma_3).$$

Then the isogenous curve $\tilde{E}$ has the Weiestrass equation $Y^2 = X^3 + \tilde{A}X + \tilde{B}$, where $\tilde{A} = A - 5t$ and $\tilde{B} = B - 7w$.

The constant σ introduced in the previous proposition is the sum of the abscissas of the points in the kernel F of I. In the important case where ℓ is odd, the non-zero points in F come into pairs $\{(x_Q, y_Q), (x_Q, -y_Q)\}$, so that we will write

$$D(x) = g(x)^2 \quad \text{with} \quad g(x) = x^{(\ell-1)/2} - q_1 x^{(\ell-3)/2} + \cdots,$$

and $\sigma = 2q_1$. Then, we can replace $D'(x)/D(x)$ by $2g'(x)/g(x)$ in Proposition 4.1.

4.2 Elkies' quadratic algorithm

From now on, we are given the two curves E and $\tilde{E}$ through their Weierstrass equations, admitting a normalized isogeny $I : E \to \tilde{E}$ of degree ℓ. We will write

$$E : \ y^2 = x^3 + Ax + B \quad \text{and} \quad \tilde{E} : \ y^2 = x^3 + \tilde{A}x + \tilde{B}.$$

From this input, and possibly that of σ, we want to determine the isogeny I, which we write as in Equation (11)

$$I(x, y) = \left(\frac{N(x)}{D(x)}, y \left(\frac{N(x)}{D(x)} \right)' \right).$$

We first describe an algorithm due to Elkies [22], that we call Elkies1998, whose complexity is quadratic in the degree ℓ. In the next subsection, we give two fast variants of algorithm Elkies1998, called fastElkies and fastElkies′, of respective complexities $O(\mathsf{M}(\ell))$ and $O(\mathsf{M}(\ell)\log\ell)$.

The algorithm Elkies1998 was introduced for the prime degree case in [22], but it works for any ℓ large enough. The first part of the algorithm aims at computing the expansion of $N(x)/D(x)$ at infinity; the second part amounts to recovering the power sums of the roots of $D(x)$ from this expansion.

To present these ideas, our starting remark is that the rational function $N(x)/D(x)$ satisfies the non-linear differential equation

$$(x^3 + Ax + B)\left(\frac{N(x)}{D(x)} \right)'^2 = \left(\frac{N(x)}{D(x)} \right)^3 + \tilde{A} \left(\frac{N(x)}{D(x)} \right) + \tilde{B}. \tag{14}$$

This follows from Proposition 4.1 and the fact that I maps E onto $\tilde{E}$. Differentiating Equation (14) leads to the following second-order equation:

$$(3x^2 + A) \left(\frac{N(x)}{D(x)} \right)' + 2(x^3 + Ax + B) \left(\frac{N(x)}{D(x)} \right)'' = 3 \left(\frac{N(x)}{D(x)} \right)^2 + \tilde{A}. \tag{15}$$

Writing the expansion of the rational function $N(x)/D(x)$ at infinity

$$\frac{N(x)}{D(x)} = x + \sum_{i \geq 1} \frac{h_i}{x^i}$$

and identifying coefficients of x^{-i} from both sides of Equation (15) yields the recurrence

$$h_k = \frac{3}{(k-2)(2k+3)} \sum_{i=1}^{k-2} h_i h_{k-1-i} - \frac{2k-3}{2k+3} A h_{k-2} - \frac{2(k-3)}{2k+3} B h_{k-3}, \quad \text{for all } k \geq 3, \tag{16}$$

with initial conditions

$$h_1 = \frac{A - \tilde{A}}{5} \quad \text{and} \quad h_2 = \frac{B - \tilde{B}}{7}.$$

The recurrence (16) is the basis of algorithm Elkies1998; using it, one can compute $h_3, \ldots, h_{\ell-2}$ using $O(\ell^2)$ operations in $\mathbf{K}$.

Elkies' algorithm Elkies1998 assumes that σ is given. Extracting coefficients in Equation (13) then yields

$$h_i = (2i+1)p_{i+1} + (2i-1)Ap_{i-1} + (2i-2)Bp_{i-2}, \quad \text{for all } i \geq 1. \tag{17}$$

Since $h_1, \ldots, h_{\ell-2}$ are known, $p_2, \ldots, p_{\ell-1}$ can be deduced from the previous recurrence using $O(\ell)$ operations. The polynomial $D(x)$ is then recovered, either by a quadratic algorithm or the faster algorithm of §2.2, and $N(x)$ is deduced using formula (13), in $O(\mathsf{M}(\ell))$ operations.

This algorithm requires that $2, \ldots, 2\ell-1$ be units in $\mathbf{K}$. Its complexity is in $O(\ell^2)$, the bottleneck being the computation of the coefficients $h_1, \ldots, h_{\ell-2}$. Observe the parallel with the computations presented in the previous section, where differentiating Weierstrass' equation yields the recurrence (7), which appears as a particular case of the recurrence (16) (the former is obtained by taking $A = B = 0$ in the latter).

4.3 Fast algorithms

We improve on the computation of the coefficients h_i in algorithm Elkies1998, the remaining part being unchanged. Unfortunately, we cannot directly apply the algorithm of §2.4 to compute the expansion of $N(x)/D(x)$ at infinity using the differential equation (14), since the equation obtained by the change of variables $x \mapsto 1/x$ is singular at the origin. To avoid this technical complication, we rather consider the power series

$$S(x) = x + \frac{\tilde{A} - A}{10} x^5 + \frac{\tilde{B} - B}{14} x^7 + O(x^9) \in x + x^3 \mathbf{K}[[x^2]]$$

such that

$$\frac{N(x)}{D(x)} = \frac{1}{S\left(\frac{1}{\sqrt{x}}\right)^2};$$

remark that S satisfies the relation $\tilde{R} = S \circ R$, with the notation $R(z) = 1/\sqrt{\wp(z)}$ and $\tilde{R}(z) = 1/\sqrt{\tilde{\wp}(z)}$ introduced in §3.3.

Applying the chain rule gives the following first order differential equation satisfied by $S(x)$:

$$(Bx^6 + Ax^4 + 1)\, S'(x)^2 = 1 + \tilde{A}\, S(x)^4 + \tilde{B}\, S(x)^6.$$

Using this differential equation, we propose two algorithms to compute $N(x)/D(x)$, depending on whether the coefficient σ is known or not. In the algorithms, we write

$$S(x) = xT(x^2) \quad \text{and} \quad U(x) = \frac{1}{T(x)^2} \in 1 + x^2\mathbf{K}[[x]] \quad \text{so that} \quad \frac{N(x)}{D(x)} = x\, U\left(\frac{1}{x}\right).$$

The first algorithm, called fastElkies, assumes that σ is known and goes as follows.

1. Compute $C(x) = (Bx^6 + Ax^4 + 1)^{-1} \bmod x^{2\ell-1} \in \mathbf{K}[[x]]$;
2. Compute $S(x) \bmod x^{2\ell}$ using the algorithm of §2.4 with $G(x,t) = C(x)(1 + \tilde{A}t^4 + \tilde{B}t^6)$, and deduce $T(x) \bmod x^{\ell}$;
3. Compute $U(x) = 1/T(x)^2 \bmod x^{\ell}$ using the algorithm in §2.1;
4. Compute the coefficients $h_1, \ldots, h_{\ell-2}$ of $N(x)/D(x)$, using $N(x)/D(x) = xU(1/x)$;
5. Compute the power sums $p_2, \ldots, p_{\ell-1}$ of $D(x)$, using the linear recurrence (17);
6. Recover $D(x)$ from its power sums, as described in §2.2;
7. Deduce $N(x)$ using Equation (13).

Steps (1) and (5) have cost $O(\ell)$. Steps (2), (3), (6) and (7) can be performed in $O(\mathsf{M}(\ell))$ operations, and Step (4) requires no operation. This proves the first part of Theorem 2.

For our second algorithm, that we call fastElkies′, we do not assume prior knowledge of σ. Its steps (1')–(3') are just a slight variation of Steps (1)–(3), of the same complexity $O(\mathsf{M}(\ell))$, up to constant factors.

(1') Compute $C(x) = (Bx^6 + Ax^4 + 1)^{-1} \bmod x^{8\ell-5} \in \mathbf{K}[[x]]$;

(2') Compute $S(x) \bmod x^{8\ell-4}$ using the algorithm of §2.4 with $G(x,t) = C(x)(1 + \tilde{A}t^4 + \tilde{B}t^6)$, and deduce $T(x) \bmod x^{4\ell-2}$;

(3') Compute $U(x) = 1/T(x)^2 \bmod x^{4\ell-2}$, using the algorithm in §2.1;

(4') Reconstruct the rational function $U(x)$;

(5') Return $N(x)/D(x) = xU(1/x)$.

Using fast rational reconstruction, Step (4') can be performed in $O(\mathsf{M}(\ell)\log\ell)$ operations in $\mathbf{K}$. Finally, it is easy to check that our algorithm fastElkies requires that $2, \ldots, 2\ell - 1$ be units in $\mathbf{K}$, while algorithm fastElkies′ requires that $2, \ldots, 8\ell - 5$ be units in $\mathbf{K}$. This completes the proof of Theorem 2.

In the case of odd ℓ, we can compute $g(x)$ instead of $D(x)$. Accordingly, we modify the recurrence relations, and compute fewer terms. Let $q_1, q_2, \ldots$ denote the power sums

of $g(x)$, so that $q_i = p_i/2$. Then, the coefficients h_i and the power sums q_i are related by the relation

$$h_i = (4i+2)q_{i+1} + (4i-2)Aq_{i-1} + (4i-4)Bq_{i-2}. \tag{18}$$

To compute $g(x)$ using algorithm fastElkies, it suffices to compute $S(x) \bmod x^{\ell+1}$; then $T(x)$ and $U(x)$ are computed modulo $x^{(\ell+1)/2}$. Similarly, in algorithm fastElkies′, it is enough to compute $S(x) \bmod x^{4\ell}$, and $T(x)$ and $U(x)$ modulo $x^{2\ell}$.

5 Examples of isogeny computations

5.1 Worked example

Since the case of ℓ odd is quite important in practice, we first give an example of such a situation (see below for an example with $\ell = 6$). Let

$$E : y^2 = x^3 + x + 1 \quad \text{and} \quad \tilde{E} : y^2 = x^3 + 75x + 16$$

be defined over $\mathbb{F}_{101}$, with $\ell = 11$ and $\sigma = 50$. Since ℓ is odd, we will compute the polynomial $g(x)$, which has degree 5. First, from the differential equation

$$(x^6 + x^4 + 1)S'(x)^2 = 1 + 75S(x)^4 + 16S(x)^6, \quad S(0) = 0, \quad S'(0) = 1$$

we infer the equalities

$$\begin{aligned}
C &= 1 + 100\,x^4 + 100\,x^6 + x^8 + 2\,x^{10} + O(x^{11}),\\
S &= x + 68\,x^5 + 66\,x^7 + 60\,x^9 + 84\,x^{11} + O(x^{12}),\\
\text{so that}\quad T &= 1 + 68\,x^2 + 66\,x^3 + 60\,x^4 + 84\,x^5 + O(x^6),\\
\text{and}\quad T^2 &= 1 + 35x^{\,2} + 31x^{\,3} + 98x^{\,4} + 54x^{\,5} + O(x^{\,6}),\\
\text{whence}\quad U &= 1 + 66x^{\,2} + 70x^{\,3} + 16x^{\,4} + 96x^{\,5} + O(x^{\,6}).
\end{aligned}$$

We deduce

$$\frac{N(x)}{D(x)} = x + \frac{66}{x} + \frac{70}{x^2} + \frac{16}{x^3} + \frac{96}{x^4} + O\left(\frac{1}{x^5}\right).$$

At this stage, we know $h_1 = 66, h_2 = 70, h_3 = 16, h_4 = 96$, as well as $q_1 = \sigma/2 = 25$. Equation (18) then writes

$$q_{i+1} = \frac{h_i - (4i-2)q_{i-1} - (4i-4)q_{i-2}}{4i+2}, \quad \text{for all } 1 \le i \le 4$$

and gives $q_2 = 43, q_3 = 91, q_4 = 86, q_5 = 63$. The main equation in §2.2 writes

$$\begin{aligned}
x^5 g\left(\frac{1}{x}\right) &= \exp_6\left(-\left(25\,x + \frac{43}{2}x^2 + \frac{91}{3}\,x^3 + \frac{86}{4}\,x^4 + \frac{63}{5}\,x^5\right)\right)\\
&= \exp_6\left(76\,x + 29\,x^2 + 37\,x^2 + 29\,x^4 + 48\,x^5\right),
\end{aligned}$$

yielding $g(x) = x^5 + 76x^4 + 89x^3 + 24x^2 + 97x + 5$. For the sake of completeness, we have:

$N(x) = x^{11} + 51x^{10} + 61x^9 + 44x^8 + 71x^7 + 39x^6 + 81x^5 + 43x^4 + 15x^3 + 5x^2 + 24x + 15.$

Had we computed the solution $S(x)$ at precision $O(x^{44})$, the expansion at infinity of $N(x)/D(x)$ would have been known at precision $O(1/x^{21})$, and this would have sufficed to recover both $N(x)$ and $D(x)$ by rational function reconstruction, without the prior knowledge of σ.

5.2 Further examples.

As it turns out, all the theory developed for the prime case in the SEA algorithm works also, *mutatis mutandis*, in the more general case of a cyclic isogeny of non prime degree. Consider the curve

$$E : y^2 = x^3 + x + 3$$

defined over $\mathbb{F}_{1009}$. For $\ell = 6$, we find that the modular polynomial of degree 6 (obtained as the resultant of the modular polynomials of degrees 2 and 3) has three roots, one of which is $\tilde{\jmath} = 248$. Using the formulas of [5], *that are still valid*, we find the isogenous curve

$$\tilde{E} : y^2 = x^3 + 830x + 82$$

and $\sigma = 739$, from which we obtain

$$\frac{N(x)}{D(x)} = \frac{x^6 + 270\,x^5 + 325\,x^4 + 566\,x^3 + 382\,x^2 + 555\,x + 203}{x^5 + 270\,x^4 + 289\,x^3 + 659\,x^2 + 533\,x + 399}.$$

The denominator factors as

$$(x - 66)\,(x - 23)^2\,(x - 818)^2\,.$$

The value $x = 66$ corresponds to one of the roots of x^3+x+3 and is therefore the abscissa of a point of 2-torsion; 23 is the abscissa of a point of 3-torsion; 818 is the abscissa of a primitive point of 6-torsion.

As an aside, let us illustrate the case of a non cyclic isogeny. The curve E happens to have rational 2-torsion; the subgroup $E[2]$ of 2-torsion is non cyclic, being isomorphic to $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z}$. The denominator $D(x)$ appearing in the isogeny $I : E \to \tilde{E} = E/E[2]$ is simply $D(x) = x^3 + x + 3$ and Equation (13) yields $N(x) = x^4 + 1007\,x^2 + 985\,x + 1$. From this, we can compute the equation of $\tilde{E}$, namely, $y^2 = x^3 + 16x + 192$.

6 A survey of previous algorithms for isogenies

In this section, we recall and give complexity results for other known algorithms for computing isogenies. In what follows, we write the Weierstrass functions $\wp$ and $\tilde{\wp}$ of our two curves E and $\tilde{E}$ as

$$\wp(z) = \frac{1}{z^2} + \sum_{i \geq 1} c_i z^{2i} \quad \text{and} \quad \tilde{\wp}(z) = \frac{1}{z^2} + \sum_{i \geq 1} \tilde{c}_i z^{2i}.$$

All algorithms below require the knowledge of the expansion of these functions at least to precision ℓ, so they only work under a hypothesis of the type $p \gg \ell$ or $p = 0$.

We can freely assume that these expansions are known. Indeed, by Theorem 1, given A, B and $\tilde{A}, \tilde{B}$, we can precompute the coefficients c_i and $\tilde{c}_i$ up to (typically) $i = \ell - 1$ using $O(\mathsf{M}(\ell))$ operations in $\mathbf{K}$, provided that the characteristic p of $\mathbf{K}$ is either 0 or $> \ell$. This turns out to be negligible compared to the other costs involved in the following algorithms.

6.1 First algorithms

A brute force approach to compute $N(x)/D(x)$ is to use the equation

$$\tilde{\wp}(z) = \wp\left(\frac{N(z)}{D(z)}\right) \tag{19}$$

and the method of undetermined coefficients. This reduces to computing $\wp(z)^i \bmod z^{4\ell-2}$ for $1 \leq i \leq \ell$ and solving a linear system with $2\ell - 1$ unknowns. This direct method requires that $2, \ldots, 4\ell$ be units in $\mathbf{K}$ and its complexity is $O(\ell^\omega)$ operations in $\mathbf{K}$, where $2 \leq \omega < 3$ is the exponent of matrix multiplication.

Another possible idea is to consider the rational functions $N(x)/D(x)$ and $\hat{N}(x)/\hat{D}(x)$ respectively associated to I and its dual $\hat{I}$, noticing that by definition,

$$\frac{N}{D} \circ \frac{\hat{N}}{\hat{D}} = \frac{\phi_\ell}{\psi_\ell^2},$$

using the notation of Equation (9). However, algorithms for directly decomposing ϕ_ℓ/ψ_ℓ^2 [55, 28, 1] lead to an expensive solution in our case, since they require factoring the division polynomial f_ℓ, of degree $\Theta(\ell^2)$. Indeed, even using the best (sub-quadratic) algorithms for polynomial factorization [32], of exponent 1.815, this would yield an algorithm for computing isogenies of degree ℓ in complexity more than cubic with respect to ℓ, which is unacceptable.

6.2 Stark's method

To the best of our knowledge, the first subcubic method for finding N and D is due to Stark [52] and amounts to expanding $\tilde{\wp}$ as a continued fraction in $\wp$, using Equation (19). The fraction N/D is approximated by p_n/q_n and the algorithm stops when the degree of q_n is $\ell - 1$, yielding D. In particular, it works for any degree isogeny. Since $\wp$ and $\tilde{\wp}$ are in $1/z^2 + \mathbf{K}[[z^2]]$, it is sufficient to work with series in $Z = z^2$.

1. $T := \tilde{\wp}(Z) + O(Z^\ell)$;
2. $n := 1$;
3. $q_0 := 1$;
4. $q_1 := 0$;
5. **while** $\deg(q_n) < \ell - 1$ **do**

   {at this point, $T(Z) = t_{-r}Z^{-r} + \cdots + t_0 + t_1 Z + \cdots + O(Z^{(\ell - \deg q_n - r) - 1})$}

   (a) $n := n + 1$;

   (b) $a_n := 0$;

   (c) **while** $r \geq 1$ **do**

   $a_n := a_n + t_{-r}z^r$;

$T := T - t_{-r}\wp^r = t_{-s}Z^{-s} + \cdots$;

$r := s$

(d) $q_n := a_n q_{n-1} + q_{n-2}$;

(e) $T := 1/T$;

6. Return $D := q_n$.

This algorithm (that we call Stark1972) requires $O(\ell)$ passes through Step (5); this bound is reached in general, with $r = 1$ at each step. The step that dominates the complexity is the computation of reciprocals in Step (5.e), with precision $2\ell - 1 - 2\deg q_n - 2r$. The sum of these operations thus costs $O(\ell \mathsf{M}(\ell))$. The multiplications in Step (5.d) can be done in time $O(\ell \mathsf{M}(\ell))$ as well (these multiplications could be done faster if needed). Since the largest degree of the polynomials a_n is bounded by $\ell - 1$, computing all powers of $\wp$ at Step (5.c) also fits within the $O(\ell \mathsf{M}(\ell))$ bound. Finally, knowing $D(x)$, the numerator $N(x)$ can be recovered in cost $O(\mathsf{M}(\ell))$ using Equation (13).

In the case where ℓ is odd and we need $g(x)$, as in the context of the SEA algorithm, we can compute it in $O(\mathsf{M}(\ell))$ operations by computing $\exp((\log D)/2)$.

To summarize, the total cost of algorithm Stark1972 is in $O(\ell \mathsf{M}(\ell))$. Remark that compared to the methods presented below, algorithm Stark1972 does not require the knowledge of σ. Remark also that, even though r will be 1 in general, the computation of the powers $\wp^r$ in Step (5.c) could be amortized in the context of the SEA algorithm.

6.3 Elkies' 1992 method

We reproduce the method given in [21], that we call Elkies1992 (see also e.g., [11, 39]). We suppose that ℓ is odd, so that $D(x) = g(x)^2$, though minor modifications below would lead to the general solution.

Differentiating twice Equation (6) yields

$$\frac{d^4\wp(z)}{dz^4} = 120\wp^3 + 72A\wp + 48B.$$

More generally, we obtain equalities of the form

$$\frac{d^{2k}\wp(z)}{dz^{2k}} = \mu_{k,k+1}\wp^{k+1} + \cdots + \mu_{k,0},$$

for some constants $\mu_{k,j}$ that satisfy the recurrence relation

$$\mu_{k+1,j} = (2j-2)(2j-1)\mu_{k,j-1} + (2j+1)(2j+2)A\mu_{k,j+1} + (2j+2)(2j+4)B\mu_{k,j+2},$$

with $\mu_{k,k+1} = (2k+1)!$. Using this recurrence relation, the coefficients $\mu_{k,j}$, for $k \leq d-1$ and $j \leq k+1$, can be computed in $O(\ell^2)$ operations in $\mathbf{K}$.

Elkies then showed how to use these coefficients to recover the power sums $q_2, \ldots, q_d$ of g, through the following equalities, holding for $k \geq 1$:

$$(2k)!(\tilde{c}_k - c_k) = 2(\mu_{k,0}q_0 + \cdots + \mu_{k,k+1}q_{k+1}).$$

Using these equalities, assuming that $q_1 = \sigma/2$ and the coefficients c_k, $\tilde{c}_k$ and $\mu_{k,j}$ are known, we can recover $q_2, \ldots, q_d$ by solving a triangular system, in complexity $O(\ell^2)$. We can then recover g using either a quadratic algorithm, or the faster algorithm of §2.2.

There remains here the question whether the triangular system giving $q_2, \ldots, q_d$ can be solved in quasi-linear time. To do so, one should exploit the structure of the triangular system, so as to avoid computing the $\Theta(\ell^2)$ constants $\mu_{k,j}$ explicitly.

6.4 Atkin's method

In [2], Atkin gave a formula enabling the computation of $D(x)$ (see also [40, Formula 6.13] and [45]) in the case where ℓ is odd. We extend it, so as to cover the case of arbitrary ℓ, this time recovering $D(x)$. The equation we use is

$$D(\wp(z)) = z^{2-2\ell} \exp(F(z)), \tag{20}$$

where

$$F(z) = -\sigma z^2 + 2\left(\sum_{k=1}^{\infty} (\ell c_k - \tilde{c}_k) \frac{z^{2k+2}}{(2k+1)(2k+2)}\right).$$

Since ℓ and the coefficients $c_k, \tilde{c}_k$ are all assumed to be known, one can deduce the series $F(z) \bmod z^\ell$, provided that σ is given. A direct method to determine $D(x)$ is then to compute the exponential of $F(z)$, and to recover the coefficients of $D(x)$ one at a time, as shown in the following algorithm, called Atkin1992. As before, we use series in $Z = z^2$.

1. Compute the series $P_i(Z) = \wp(Z)^i$ at order ℓ, for $1 \leq i \leq \ell - 1$;
2. Compute $G(Z) = \exp_\ell(F(Z))$;
3. $T := G$;
4. $D := 0$;
5. **for** $i := \ell - 1$ **downto** 0 **do**

   {at this point, $T = tZ^{-i} + \cdots$ }

   (a) $D := D + tz^i$;

   (b) $T := T - tP_i$.

Step (1) uses $O(\ell \mathsf{M}(\ell))$ operations; the cost of Step (2) is negligible, using either classical or fast exponentiation. Then, each pass through Step (5) costs $O(\ell)$ more operations, for a total of $O(\ell^2)$. Thus, the total cost of this algorithm is in $O(\ell \mathsf{M}(\ell))$. If this algorithm is used in the context of the SEA algorithm, Step (1) can be amortized, since it depends on E only. Therefore, all the powers of $\wp$ should be computed for the maximal value of ℓ to be used, and stored. Hence, the cost of this algorithm would be dominated by that of Step (5), yielding a method of complexity $O(\ell^2)$.

A better algorithm for computing $D(x)$, avoiding the computation of all powers of $\wp(z)$, is based on the remark that Equation (20) rewrites

$$D\left(\frac{1}{x}\right) = \mathcal{I}^{2-2\ell}\left((\exp \circ F) \circ \mathcal{I}\right), \tag{21}$$

with $\mathcal{I}(x) = \wp^{-1}(1/x)$, where $\wp^{-1}$ is the functional inverse of $\wp$. The expansion of $\mathcal{I}(x)$ at order $\Theta(\ell)$ can be computed in $O(\ell)$ operations using the differential equation

$$\mathcal{I}'(x)^2 = \frac{1}{4x(1+Ax^2+Bx^3)} \quad \text{or} \quad \mathcal{I}'(x) = \frac{1}{2\sqrt{x}} \frac{1}{\sqrt{1+Ax^2+Bx^3}}. \tag{22}$$

A linear differential equation follows:

$$\frac{\mathcal{I}'(x)}{\mathcal{I}(x)} = -\frac{1+3Ax^2+4Bx^3}{2x(1+Ax^2+Bx^3)}.$$

From there follows a linear differential equation for $\mathcal{J}(x) = x^{-1/2}\mathcal{I}(x) = \sum_{i\geq 0} a_i x^i$. Extracting coefficients in this equation then gives a linear recurrence

$$a_{i+1} = -\frac{2i-1}{2(i+1)(2i+3)}\left((2i-3)Ba_{i-2} + 2Aia_{i-1}\right) \quad \text{for } i \geq 2, \tag{23}$$

with initial conditions $a_0 = 1, a_1 = 0, a_2 = -\frac{A}{10}$. This yields the following algorithm, called AtkinModComp:

1. Compute $G(Z) = \exp_\ell(F(Z))$;
2. Compute $\mathcal{I}(x)$ using Equation (23);
3. Compute $G(\mathcal{I})$ by modular composition (which is possible since G is in $\mathbf{K}[[Z]] = \mathbf{K}[[x^2]]$);
4. Deduce D using Equation (21).

The cost of the algorithm is dominated by the composition of the series $G = \exp \circ F$ and $\mathcal{I}$. From §2.5, this can be done in $O(\mathsf{M}(\ell)\sqrt{\ell} + \ell^{\frac{\omega+1}{2}})$ or $O(\mathsf{M}(\ell)\sqrt{\ell \log \ell})$ operations in $\mathbf{K}$.

To do even better, it is fruitful to reconsider the series $G(\mathcal{I}) = (\exp \circ F) \circ \mathcal{I}$ used above, but rewriting it as $\exp \circ (F \circ \mathcal{I})$ instead; this change of point of view reveals close connections with our fastElkies algorithm. More precisely, Atkin's Equation (20) can be rewritten as

$$D(\wp(x)) = \exp\left(-\sigma x^2 + 2\iint \ell\wp(x) - \tilde{\wp}(x)\right).$$

We can then obtain $D(1/x)$ as the following exponential:

$$\begin{aligned} D\left(\frac{1}{x}\right) &= \exp\left(-\sigma\mathcal{I}^2 + 2\int \mathcal{I}' \int \mathcal{I}' \left(\frac{\ell}{x} - (\tilde{\wp}\circ\mathcal{I})(x)\right)\right) && (24)\\ &= \exp\left(-\sigma\mathcal{I}^2 + 2\int \mathcal{I}' \int \mathcal{I}' \left(\frac{\ell}{x} - \frac{N(1/x)}{D(1/x)}\right)\right) && (25)\\ &= \exp\left(-\sigma\mathcal{I}^2 + 2\int \mathcal{I}' \int \mathcal{I}' \left(\frac{\ell}{x} - \frac{1}{S(\sqrt{x})^2}\right)\right). && (26) \end{aligned}$$

Then, working out the details, the sequence of operations necessary to evaluate this exponential turns out to be the same as the one used in our algorithm fastElkies of §4.3.

This does not come as a surprise: the relation (17) used in our algorithm follows from formula (13), which can be rewritten as

$$\frac{N(x)}{D(x)} = \ell x - \sigma - 2\sqrt{x^3 + Ax + B}\left(\sqrt{x^3 + Ax + B}\,\frac{D'(x)}{D(x)}\right)'.$$

Then, Equation (25) is nothing but an integral reformulation of this last equation, taking into account the fact that $\mathcal{I}$ satisfies the differential equation (22).

6.5 Summary

In Table 1 we gather the various algorithms discussed in this article, and compare these algorithms from two points of view: their complexity (expressed in number of operations in the base field $\mathbf{K}$) and their need for σ as input.

algorithm	complexity	need of σ
linear algebra	$O(\ell^\omega)$	no
Stark1972	$O(\ell\mathsf{M}(\ell))$	no
Atkin1992	$O(\ell\mathsf{M}(\ell))$	yes
AtkinModComp	$O(\mathsf{M}(\ell)\sqrt{\ell} + \ell^{\frac{\omega+1}{2}})$ or $O(\mathsf{M}(\ell)\sqrt{\ell\log\ell})$	yes
Elkies1992	$O(\ell^2)$	yes
Elkies1998	$O(\ell^2)$	yes
fastElkies	$O(\mathsf{M}(\ell))$	yes
fastElkies′	$O(\mathsf{M}(\ell)\log\ell)$	no

Table 1: Comparison of the algorithms

7 Implementation and benchmarks

We implemented our algorithms using the NTL C++ library [46, 47] and ran the program on an AMD 64 Processor 3400+ (2.4GHz). We begin with timings for computing the expansion of $\wp$, obtained over the finite field $\mathbb{F}_{10^{2004}+4683}$; they are given in Figure 1.

The shape of both curves indicates that the theoretical complexities – quadratic vs. nearly linear – are well respected in our implementation (note that the abrupt jumps at powers of 2 reflect the performance of NTL's FFT implementation of polynomial arithmetic). Moreover, the threshold beyond which our algorithm becomes useful over the quadratic one is reasonably small, making it interesting in practice very early.

We now turn our attention to the pure isogeny part, concentrating on the case where ℓ is prime, in the context of the SEA algorithm. Hence, in this case, it suffices to compute the polynomial $g(x)$ such that $D(x) = g(x)^2$. All algorithms can be adapted to make advantage of this simplification, as exemplified in §4.3 for our algorithms fastElkies and fastElkies′.

The first series of timings concerns the computation of isogenies over a small field, $\mathbf{K} = \mathbb{F}_{10^{19}+51}$, for the curve $E : y^2 = x^3 + 4589x + 91128$. We compare in Figure 2 the performances of the algorithms Elkies1992 from §6.3 and Elkies1998 from §4.2 for isogenies

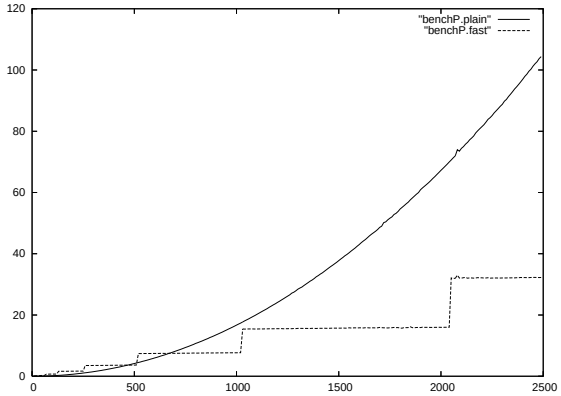

Figure 1: Timings for computing $\wp$ on $E : y^2 = x^3 + 4589x + 91128$ over $\mathbb{F}_{10^{2004}+4683}$

of moderate degree $\ell \leq 400$. Figure 3 compares the timings obtained with the algorithm Elkies1998 and our fast version fastElkies from §4.3, for isogenies of degree up to 6000.

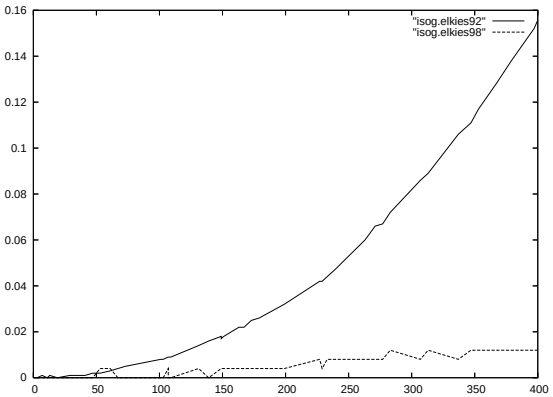

Figure 2: Elkies1992 vs. Elkies1998.

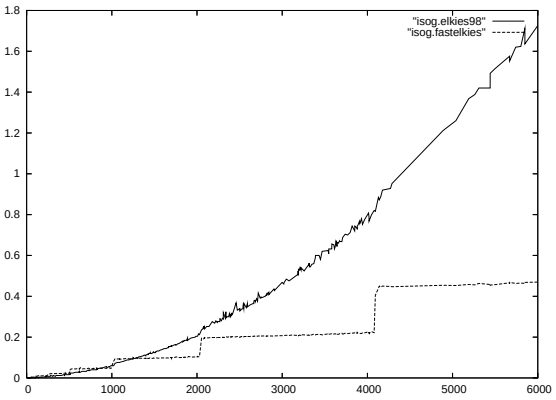

Figure 3: Elkies1998 vs. fastElkies.

Next, we compare in Figure 4 the timings obtained by the $O(\mathsf{M}(\ell))$ algorithm fastElkies, that requires the knowledge of σ, to those obtained by its $O(\mathsf{M}(\ell)\log\ell)$ counterpart fastElkies′, that does not require this information.

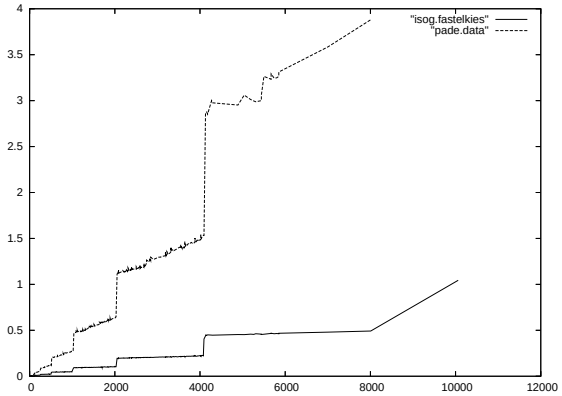

Figure 4: FastElkies vs. FastElkies′

In all figures, the degrees ℓ of the isogenies are represented on the horizontal axis and the timings are given (in seconds) on the vertical axis. Again, the shape of both curves in Figure 3 shows that the theoretical complexities are well respected in our implementation. The curves in Figure 4 show that the theoretical ratio of $\log \ell$ between algorithms fastElkies and fastElkies′ has a consequent practical impact.

Next, in Tables 2 to 8, we give detailed timings on computing ℓ-isogenies for the curve

$$E : y^2 = x^3 + Ax + B$$

where

$$A = \lfloor 10^{1990}\pi \rfloor = 31415926 \ldots 58133904,$$
$$B = \lfloor 10^{1990}e \rfloor = 27182818 \ldots 94787610,$$

for a few values of ℓ, over the larger finite field $\mathbb{F}_{10^{2004}+4683}$, and using various methods: algorithms Elkies1992, Elkies1998 and our fast variant fastElkies, Stark's algorithm Stark1972 and the two versions Atkin1992 and AtkinModComp of Atkin's algorithm.

Tables 2 and 3 give timings for basic subroutines shared by some or all of the algorithms discussed. Table 2 gives the timings necessary to compute the expansions of $\wp$ and $\tilde{\wp}$, using either the classical algorithm or our faster variant: this is used in all algorithms, except our fastElkies algorithm. Table 3 gives timings for recovering g from its power sums, first using the classical quadratic algorithm, and then using fast exponentiation as described in §2.2. This is used in algorithms Elkies1992, and Elkies1998 and its variants.

Tables 4 and 5 give the timings for algorithms Elkies1992 on the one hand and Elkies1998 and our variation fastElkies on the other hand. In Table 4, the columns μ and p_i give the time used to compute the coefficients $\mu_{i,j}$ and the power sums p_i. In Table 5, the column h_i indicates the time used to compute the coefficients h_i of the rational function N/D, first using the original quadratic algorithm Elkies1998, then using our faster variant fastElkies. The next column gives the time used to compute the power sums p_i from the h_i using recurrence (18).

Tables 6 and 7 give timings for our implementation of Atkin's original algorithm Atkin1992, as well as the faster version AtkinModComp using modular composition mentioned in §6.4. In Table 6, the column "exponential" compares the computation of $\exp(F)$

ℓ	Computing $\wp$ and $\tilde{\wp}$		
	order	quadratic	fast
1013	511	8.6	7.0
2039	1024	34.6	29.9
3019	1514	75.7	30.3
4001	2005	132.7	31
5021	2515	209.3	64.4

Table 2: Computing $\wp$ and $\tilde{\wp}$

ℓ	Recovering g	
	quadratic	fast
1013	4.2	1.1
2039	17.4	2.5
3019	38.2	5.1
4001	66.9	5.5
5021	106.2	11.2

Table 3: Recovering g from its power sums

ℓ	Elkies1992			
	$\wp, \tilde{\wp}$	μ	p_i	g
1013		10.4	4.4	
2039	See	49.1	17.9	See
3019	Table 2	130.6	38.9	Table 3
4001		263	68.4	
5021		496.5	106.6	

Table 4: Algorithm Elkies1992

ℓ	Elkies1998 and fastElkies			
	h_i		p_i	g
	quadratic	fast		
1013	4.4	4.5	0.05	
2039	17.3	9.6	0.1	See
3019	38.0	19.5	0.16	Table 3
4001	67.2	20.0	0.21	
5021	105.0	40.7	0.27	

Table 5: Algorithms Elkies1998 and fastElkies

using the naive exponentiation algorithm to the computation using the faster algorithm presented in §2.2; the column $\wp^k$ gives the time for computing all the series $\wp(z)^k$ and the column g that for recovering the coefficients of g from its power sums. Table 7 gives timings obtained using the two modular composition algorithms mentioned in §2.5, called here ModComp1 and ModComp2; the previous columns give the time for computing $\exp(F)$ and that for computing the requested power of $\mathcal{I}$; the last column gives the time to perform the final multiplication.

Asymptotically, algorithm ModComp2 is faster than algorithm ModComp1, so that the timings in Table 7 might come as a surprise. The explanation is that, for the problem sizes we are interested in, the predominant step of algorithm ModComp1 is the one based on polynomial operations, while the step based on linear algebra operations takes only about 10% of the whole computing time. Thus, the practical complexity of this algorithm in the considered range ($1000 < \ell < 6000$) is proportional to $\mathsf{M}(\ell)\sqrt{\ell}$, while that of algorithm ModComp2 is proportional to $\mathsf{M}(\ell)\sqrt{\ell \log \ell}$. Moreover, the proportionality constant is smaller in the built-in NTL function performing ModComp1 than in our implementation

ℓ	Algorithm Aktin1992				
	$\wp, \tilde{\wp}$	exponential		$\wp^k$	g
		naive	fast		
1013		88.4	1.2	72.3	4.4
2039	See	370.1	4.9	304.9	17.7
3019	Table 2	955.9	5.1	755.8	38.9
4001		1503	5.2	1218.9	67.6
5021		3180	10.8	2506.4	108.7

Table 6: Atkin's original algorithm, variations for $\exp(F)$

ℓ	Algorithm AtkinModComp					
	$\wp, \tilde{\wp}$	$\exp(F)$	$\mathcal{I}^{1-\ell}$	modular composition		g
				ModComp1	ModComp2	
1013		1.2	2.7	14.3	35.6	0.2
2039	See	2.5	6.6	45.8	111.9	0.4
3019	Table 2	5.1	10.4	95.3	241	0.7
4001		5.2	11.6	143.2	338	0.9
5021		10.9	20.9	240	642	1.4

Table 7: Atkin's algorithm with modular composition

of ModComp2.

Notice that in all the columns labelled "fast" in Tables 2–7, the timings reflect the already mentioned (piecewisely almost constant) behaviour of the FFT: polynomial multiplication in the degree range 1024–2047 is roughly twice as fast as in the range 2047–4095 and roughly four times as fast as in the range 4096–8191.

Finally, Table 8 gives timings for Stark's algorithm Stark1972; apart from the common computation of $\wp$ and $\tilde{\wp}$, we distinguish the time necessary to compute all inverses (the quadratic algorithm when available, followed by that using fast inversion) and that for deducing the polynomials q_n.

ℓ	$\wp, \tilde{\wp}$	Inverses		q_n
		quadratic	fast	
1013		23542	1222.7	28.0
2039	See	> 100000	5113.4	116.9
3019	Table 2		12182	258
4001			20388	418.6
5021			38910	663.1

Table 8: Stark's algorithm Stark1972

Conclusion

The complexity analyses of the algorithms we have surveyed shows that for the case of a large prime characteristic and for a reasonably large degree ℓ of the isogeny, our new $O(\mathsf{M}(\ell))$ algorithm improves over previously known techniques.

The current implementation of our algorithm can be further optimized to make it the algorithm of choice for smaller values of the degree. Indeed, it is known that algorithms based on Newton iteration present certain redundancies (coefficients that can be predicted in advance, repeated multiplicands). Removing these redundancies is feasible (see [4, 29]), allowing one to achieve constant-factor speed-ups. For the moment, our implementation relies only partially on these techniques; we believe that further programming effort would bring practical improvements by non-negligible constant factors.

Another direction for future work is to adapt our ideas to the case of a small characteristic. In this respect, modifying the last phase of the algorithm of Joux and Lercier [31] seems a promising search path.

Acknowledgments. We thank Pierrick Gaudry for his remarks during the elaboration of the ideas contained in this work.

References

[1] C. Alonso, J. Gutierrez, and T. Recio. A rational function decomposition algorithm by near-separated polynomials. *Journal of Symbolic Computation*, 19(6):527–544, 1995.

[2] A. O. L. Atkin. The number of points on an elliptic curve modulo a prime (II). Draft. Available at `http://listserv.nodak.edu/archives/nmbrthry.html`, July 1992.

[3] D. J. Bernstein. Composing power series over a finite ring in essentially linear time. *Journal of Symbolic Computation*, 26(3):339–341, 1998.

[4] D. J. Bernstein. Removing redundancy in high-precision Newton iteration, 2000. Available on-line at `http://cr.yp.to/fastnewton.html`.

[5] I. Blake, G. Seroussi, and N. Smart. *Elliptic curves in cryptography*, volume 265 of *London Mathematical Society Lecture Notes Series*. Cambridge University Press, 1999.

[6] R. P. Brent. Multiple-precision zero-finding methods and the complexity of elementary function evaluation. In *Analytic computational complexity*, pages 151–176. Academic Press, New York, 1976. Proceedings of a Symposium held at Carnegie-Mellon University, Pittsburgh, Pa., 1975.

[7] R. P. Brent, F. G. Gustavson, and D. Y. Y. Yun. Fast solution of Toeplitz systems of equations and computation of Padé approximants. *Journal of Algorithms*, 1(3):259–295, 1980.

[8] R. P. Brent and H. T. Kung. Fast algorithms for manipulating formal power series. *Journal of the ACM*, 25(4):581–595, 1978.

[9] E. Brier and M. Joye. Fast point multiplication on elliptic curves through isogenies. In *Applied algebra, algebraic algorithms and error-correcting codes (Toulouse, 2003)*, volume 2643 of *Lecture Notes in Computer Science*, pages 43–50. Springer, Berlin, 2003.

[10] D. G. Cantor and E. Kaltofen. On fast multiplication of polynomials over arbitrary algebras. *Acta Informatica*, 28(7):693–701, 1991.

[11] L. S. Charlap, R. Coley, and D. P. Robbins. Enumeration of rational points on elliptic curves over finite fields. Draft, 1991.

[12] S. Cook. *On the minimum computation time of functions.* PhD thesis, Harvard University, 1966.

[13] J.-M. Couveignes. *Quelques calculs en théorie des nombres.* Thèse, Université de Bordeaux I, July 1994.

[14] J.-M. Couveignes. Computing l-isogenies using the p-torsion. In H. Cohen, editor, *Algorithmic Number Theory*, volume 1122 of *Lecture Notes in Computer Science*, pages 59–65. Springer-Verlag, 1996. Proceedings of the Second International Symposium, ANTS-II, Talence, France, May 1996.

[15] J.-M. Couveignes. Isomorphisms between Artin-Schreier towers. *Mathematics of Computation*, 69(232):1625–1631, 2000.

[16] J.-M. Couveignes, L. Dewaghe, and F. Morain. Isogeny cycles and the Schoof-Elkies-Atkin algorithm. Research Report LIX/RR/96/03, LIX, April 1996. Available at `http://www.lix.polytechnique.fr/Labo/Francois.Morain/`.

[17] J.-M. Couveignes and T. Henocq. Action of modular correspondences around CM points. In C. Fieker and D. R. Kohel, editors, *Algorithmic Number Theory*, volume 2369 of *Lecture Notes in Computer Science*, pages 234–243. Springer-Verlag, 2002. Proceedings of the 5th International Symposium, ANTS-V, Sydney, Australia, July 2002.

[18] J.-M. Couveignes and F. Morain. Schoof's algorithm and isogeny cycles. In L. Adleman and M.-D. Huang, editors, *Algorithmic Number Theory*, volume 877 of *Lecture Notes in Computer Science*, pages 43–58. Springer-Verlag, 1994. 1st Algorithmic Number Theory Symposium - Cornell University, May 6-9, 1994.

[19] L. Dewaghe. Isogénie entre courbes elliptiques. *Utilitas Mathematica*, 55:123–127, 1999.

[20] C. Doche, T. Icart, and D. R. Kohel. Efficient scalar multiplication by isogeny decompositions. Cryptology ePrint Archive, Report 2005/420, 2005. `http://eprint.iacr.org/`.

[21] N. D. Elkies. Explicit isogenies. Draft, 1992.

[22] N. D. Elkies. Elliptic and modular curves over finite fields and related computational issues. In D. A. Buell and J. T. Teitelbaum, editors, *Computational Perspectives on Number Theory: Proceedings of a Conference in Honor of A. O. L. Atkin*, volume 7 of *AMS/IP Studies in Advanced Mathematics*, pages 21–76. American Mathematical Society, International Press, 1998.

[23] M. Fouquet and F. Morain. Isogeny volcanoes and the SEA algorithm. In C. Fieker and D. R. Kohel, editors, *Algorithmic Number Theory*, volume 2369 of *Lecture Notes in Computer Science*, pages 276–291. Springer-Verlag, 2002. Proceedings of the 5th International Symposium, ANTS-V, Sydney, Australia, July 2002.

[24] S. Galbraith. Constructing isogenies between elliptic curves over finite fields. *Journal of Computational Mathematics*, 2:118–138, 1999.

[25] S. D. Galbraith, F. Hess, and N. P. Smart. Extending the GHS Weil descent attack. In *Advances in cryptology—EUROCRYPT 2002 (Amsterdam)*, volume 2332 of *Lecture Notes in Computer Science*, pages 29–44. Springer, Berlin, 2002.

[26] J. von zur Gathen and J. Gerhard. *Modern computer algebra*. Cambridge University Press, 1999.

[27] H. Gunji. The Hasse invariant and p-division points of an elliptic curve. *Arch. Math.*, 27(2):148–158, 1976.

[28] J. Gutierrez and T. Recio. A practical implementation of two rational function decomposition algorithms. In *Proceedings ISSAC'92*, pages 152–157. ACM, 1992.

[29] G. Hanrot, M. Quercia, and P. Zimmermann. The middle product algorithm, I. Speeding up the division and square root of power series. *Applicable Algebra in Engineering, Communication and Computing*, 14(6):415–438, 2004.

[30] D. Jao, S. D. Miller, and R. Venkatesan. Do all elliptic curves of the same order have the same difficulty of discrete log? In Bimal Roy, editor, *Advances in Cryptology – ASIACRYPT 2005*, volume 3788 of *Lecture Notes in Computer Science*, pages 21–40, 2005. 11th International Conference on the Theory and Application of Cryptology and Information Security, Chennai, India, December 4-8, 2005.

[31] A. Joux and R. Lercier. Counting points on elliptic curves in medium characteristic. Cryptology ePrint Archive, Report 2006/176, 2006. `http://eprint.iacr.org/`.

[32] E. Kaltofen and V. Shoup. Subquadratic-time factoring of polynomials over finite fields. *Mathematics of Computation*, 67(223):1179–1197, 1998.

[33] K. S. Kedlaya. Counting points on hyperelliptic curves using Monsky-Washnitzer cohomology. *Journal of the Ramanujan Mathematical Society*, 16(4):323–338, 2001.

[34] D. Kohel. *Endomorphism rings of elliptic curves over finite fields*. PhD thesis, University of California at Berkeley, 1996.

[35] H. T. Kung. On computing reciprocals of power series. *Numerische Mathematik*, 22:341–348, 1974.

[36] R. Lercier. Computing isogenies in F_{2^n}. In H. Cohen, editor, *Algorithmic Number Theory*, volume 1122 of *Lecture Notes in Computer Science*, pages 197–212. Springer Verlag, 1996. Proceedings of the Second International Symposium, ANTS-II, Talence, France, May 1996.

[37] R. Lercier. *Algorithmique des courbes elliptiques dans les corps finis.* Thèse, École polytechnique, June 1997.

[38] R. Lercier and F. Morain. Computing isogenies between elliptic curves over F_{p^n} using Couveignes's algorithm. *Mathematics of Computation*, 69(229):351–370, January 2000.

[39] F. Morain. Calcul du nombre de points sur une courbe elliptique dans un corps fini : aspects algorithmiques. *Journal de Théorie des Nombres de Bordeaux*, 7(1):255–282, 1995.

[40] V. Müller. *Ein Algorithmus zur Bestimmung der Punktanzahl elliptischer Kurven über endlichen Körpern der Charakteristik größer drei.* PhD thesis, Technischen Fakultät der Universität des Saarlandes, 1995.

[41] A. Rostovtsev and A. Stolbunov. Public-key cryptosystem based on isogenies. Cryptology ePrint Archive, Report 2006/145, 2006. `http://eprint.iacr.org/`.

[42] T. Satoh. The canonical lift of an ordinary elliptic curve over a finite field and its point counting. *Journal of the Ramanujan Mathematical Society*, 15:247–270, 2000.

[43] A. Schönhage. The fundamental theorem of algebra in terms of computational complexity. Technical report, Mathematisches Institut der Universität Tübingen, 1982. Preliminary report.

[44] A. Schönhage and V. Strassen. Schnelle Multiplikation großer Zahlen. *Computing*, 7:281–292, 1971.

[45] R. Schoof. Counting points on elliptic curves over finite fields. *Journal de Théorie des Nombres de Bordeaux*, 7(1):219–254, 1995.

[46] V. Shoup. A new polynomial factorization algorithm and its implementation. *Journal of Symbolic Computation*, 20(4):363–397, 1995.

[47] V. Shoup. *The Number Theory Library.* 1996–2005. `http://www.shoup.net/ntl`.

[48] M. Sieveking. An algorithm for division of powerseries. *Computing*, 10:153–156, 1972.

[49] J. H. Silverman. *The arithmetic of elliptic curves*, volume 106 of *Graduate Texts in Mathematics*. Springer, 1986.

[50] J. H. Silverman. *Advanced topics in the arithmetic of elliptic curves*, volume 151 of *Graduate Texts in Mathematics*. Springer, 1994.

[51] N. P. Smart. An analysis of Goubin's Refined Power Analysis Attack. In *Cryptographic Hardware and Embedded Systems – CHES 2003*, volume 2779 of *Lecture Notes in Computer Science*, pages 281–290, Berlin, 2003. Springer.

[52] H. M. Stark. Class-numbers of complex quadratic fields. In W. Kuyk, editor, *Modular functions of one variable I*, volume 320 of *Lecture Notes in Mathematics*, pages 155–174. Springer Verlag, 1973. Proceedings International Summer School University of Antwerp, RUCA, July 17-Agust 3, 1972.

[53] E. Teske. An elliptic trapdoor system. *Journal of Cryptology*, 19(1):115–133, 2006.

[54] J. Vélu. Isogénies entre courbes elliptiques. *Comptes-Rendus de l'Académie des Sciences, Série I*, 273:238–241, juillet 1971.

[55] R. Zippel. Rational function decomposition. In Stephen M. Watt, editor, *Symbolic and algebraic computation*, pages 1–6, New York, 1991. ACM Press. Proceedings of ISSAC'91, Bonn, Germany.